Ex2b

In this exercise you will implement the bellman ford algorithm for routing.
Your program will simulate a router, i.e., if there are n routers, you should run it n times in n different terminals.
The input to your program is a file that contains the routers information, the name of the router and number of trials to connect to a neighbor router (more on this later).
Here is an example of the input file:

```
3
Router1 127.0.0.1 7000
Router2 127.0.0.1 8000
Router3 127.0.0.1 9000
Router1 Router2 1
Router1 Router3 2
Router2 Router3 7
```

3 is the number of routers, and then there is a line for each router with its name, ip and port and then the edges: source, destination and weight.

In the first stage, each router r1 updates its distance vector to be 0 for r1, c(r1,r2) which is the cost of the edge (r1,r2) if r2 is a neighbor and infinity otherwise.

Each router creates 2 threads for each neighbor, one to send data and one to receive data and additional thread that re-compute the DV as needed.
All the connection should be through TCP sockets (much more simple than UDP).

Each sender thread should create a socket and sends its DV to the neighbor, note that a router should send its DV only if the DV was changed (hint: conditional variable).

Each receiver thread should create socket and try to read the neighbor's DV, if the neighbor didn't change its dv, it sends the message "0" (as an int), if it did change it, it sends "1DV", i.e., the int 1 and then the distance vector. When the router gets the DV (or 0) from all its neighbors, it recompute its DV.

The last thread is waked up (hint: conditional variable) when the router gets messages from all of its neighbors. Then it checks if there was a change in one of the DV of its neighbors, if not, the program should print the DV and exits, if yes, the DV should recomputed (relaxing by all neighbors) and a message should be sent to the neighbors, either 0 if the DV wasn't changed or 1DV otherwise.

General Pseudo:

<u>Main:</u>
Read file, initialized data structure.
Initialized own DV and via
Create threads
Join threads

<u>Sender to neighbor i:</u>
Create socket
The port to connect is calculated as follows: port of neighbor (from file) + sum of ascii values of characters in the name of the source. For example, if abc sends to xyz and the port of xyz is 6000, then the port is 6000+97+98+99. You can assume that the ports will not overlap each other.

Try to connect to neighbor.
Since we cannot run all terminals at the same time, we must consider a case where connect fails. Run connect in a loop argv[3] times, after each failure, sleep 1 second. You can assume that by the end of this process you succeed to connect all neighbors.

Then run in an endless loop:

```
If own dv was not changed,
    If the calculation of the dv is done you can finalize sender.
    Otherwise, wait (conditional variable)

If dv was change d
Send 1|dv (| for concatenation, 1 as an integer)
Else
Send 0
```

<u>Receiver from neighbor i:</u>
Create socket
Accept connection on the following port: source port + sum of ascii values of the neighbor name.
Run in endless loop:

```
If the dv calculation process is done, finalize receiver
(Done means that the source has the most updated vector, i.e. it got 0 from all its neighbors)

If got 0
    DV of Neighbor i doesn't change
Else
    Update DV for neighbor i
If got DV or 0 from all neighbors
Wake calculator
```

<u>Calculator:</u>

```
If not got all Dvs or 0 (conditional variable)
Wait

Relax (input is DV of own and neighbors)
Wake sender

If got 0 from all
Print routing table (same format as in ex2a)
Clean and exit
```

Error handling:
If some of the system calls fails, don't kill the whole program, just the specific thread.

You can assume that there are no negative weights.

Good luck!